

# Growing perfect cubes

Márk Horváth, Antal Iványi\*

*Eötvös Loránd University, Department of Computer Algebra H-1117 Budapest, Pázmány Péter sétány 1/c*

Received 5 June 2006; accepted 8 August 2007

Available online 27 September 2007

Dedicated to the 60th birthday of Miklós Simonovits

---

## Abstract

An  $(n, a, b)$ -perfect double cube is a  $b \times b \times b$  sized  $n$ -ary periodic array containing all possible  $a \times a \times a$  sized  $n$ -ary array exactly once as subarray. A growing cube is an array whose  $c_j \times c_j \times c_j$  sized prefix is an  $(n_j, a, c_j)$ -perfect double cube for  $j = 1, 2, \dots$ , where  $c_j = n_j^{v/3}$ ,  $v = a^3$  and  $n_1 < n_2 < \dots$ . We construct the smallest possible perfect double cube (a  $256 \times 256 \times 256$  sized 8-ary array) and growing cubes for any  $a$ .

© 2007 Elsevier B.V. All rights reserved.

*MSC:* 05B15; 68R05; 94A55

*Keywords:* de Bruijn array; Perfect map; Colouring

---

## 1. Introduction

Cyclic sequences in which every possible sequence of a fixed length occurs exactly once have been studied for more than a hundred years [6]. The same problem, which can be applied to position localization, was extended to arrays [5].

Let  $\mathbb{Z}$  be the set of integers. For  $u, v \in \mathbb{Z}$  we denote the set  $\{j \in \mathbb{Z} \mid u \leq j \leq v\}$  by  $[u..v]$  and the set  $\{j \in \mathbb{Z} \mid j \geq u\}$  by  $[u..\infty]$ . Let  $d \in [1..\infty]$  and  $k, n \in [2..\infty]$ ,  $b_i, c_i, j_i \in [1..\infty]$  ( $i \in [1..d]$ ) and  $a_i, k_i \in [2..\infty]$  ( $i \in [1..d]$ ). Let  $\mathbf{a} = \langle a_1, a_2, \dots, a_d \rangle$ ,  $\mathbf{b} = \langle b_1, b_2, \dots, b_d \rangle$ ,  $\mathbf{c} = \langle c_1, c_2, \dots, c_d \rangle$ ,  $\mathbf{j} = \langle j_1, j_2, \dots, j_d \rangle$  and  $\mathbf{k} = \langle k_1, k_2, \dots, k_d \rangle$  be vectors of length  $d$ ,  $\mathbf{n} = \langle n_1, n_2, \dots \rangle$  an infinite vector with  $2 \leq n_1 < n_2 < \dots$ .

**Definition 1.** A  $d$  dimensional  $n$ -ary array  $A$  is a mapping  $A : [1..\infty]^d \rightarrow [0, n - 1]$ . If there exist a vector  $\mathbf{b}$  and an array  $M$  such that

$$\forall \mathbf{j} \in [1..\infty]^d : A[\mathbf{j}] = M[(j_1 \bmod b_1) + 1, (j_2 \bmod b_2) + 1, \dots, (j_d \bmod b_d) + 1],$$

then  $A$  is a  $\mathbf{b}$ -periodic array and  $M$  is a period of  $A$ . The  $\mathbf{a}$ -sized subarrays of  $A$  are the  $\mathbf{a}$ -periodic  $n$ -ary arrays.

Although our arrays are infinite we say that a  $\mathbf{b}$ -periodic array is  $\mathbf{b}$ -sized.

---

\* Corresponding author.

*E-mail addresses:* [cyber@inf.elte.hu](mailto:cyber@inf.elte.hu) (M. Horváth), [tony@compalg.inf.elte.hu](mailto:tony@compalg.inf.elte.hu) (A. Iványi)

*URLs:* <http://people.inf.elte.hu/cyber>, <http://people.inf.elte.hu/tony> (A. Iványi).

**Definition 2.** Indexset  $A_{index}$  of a  $\mathbf{b}$ -periodic array  $A$  is the Cartesian product

$$A_{index} = \times_{i=1}^d [1..b_i].$$

**Definition 3.** A  $d$  dimensional  $\mathbf{b}$ -periodic  $n$ -ary array  $A$  is called  $(n, d, \mathbf{a}, \mathbf{b})$ -perfect, if all possible  $n$ -ary arrays of size  $\mathbf{a}$  appear in  $A$  exactly once as a subarray.

Here  $n$  is the alphabet size,  $d$  gives the number of dimensions of the “window” and the perfect array  $M$ , the vector  $\mathbf{a}$  characterizes the size of the window, and the vector  $\mathbf{b}$  is the size of the perfect array  $M$ .

**Definition 4.** An  $(n, d, \mathbf{a}, \mathbf{b})$ -perfect array  $A$  is called  $\mathbf{c}$ -cellular, if  $c_i$  divides  $b_i$  for  $i \in [1..d]$ . A cellular array consists of  $b_1/c_1 \times b_2/c_2 \times \cdots \times b_d/c_d$  disjoint subarrays of size  $\mathbf{c}$ , called *cells*. In each cell the element with smallest indices is called the *head* of the cell. The contents of the cell is called *pattern*.

**Definition 5.** The product of the elements of a vector  $\mathbf{a}$  is called the *volume* of the vector and is denoted by  $|\mathbf{a}|$ . The number of elements of perfect array  $M$  is called the volume of  $M$  and is denoted by  $|M|$ .

**Definition 6.** If  $b_1 = b_2 = \cdots = b_d$ , then the  $(n, d, \mathbf{a}, \mathbf{b})$ -perfect array  $A$  is called *symmetric*. If  $A$  is symmetric and  $a_1 = a_2 = \cdots = a_d$ , then  $A$  is called *doubly symmetric*. If  $A$  is doubly symmetric and

- (1)  $d = 1$ , then  $A$  is called a *double sequence*;
- (2)  $d = 2$ , then  $A$  is called a *double square*;
- (3)  $d = 3$ , then  $A$  is called a *double cube*.

According to this definition, all perfect sequences are doubly symmetric. In the case of symmetric arrays we use the notion  $(n, d, \mathbf{a}, b)$  and in the case of doubly symmetric arrays we use  $(n, d, a, b)$  instead of  $(n, d, \mathbf{a}, \mathbf{b})$ .

The first known result originates from Flye-Sainte [6] who proved the existence of  $(2, 1, a, 2^a)$ -perfect sequences for all possible values of  $a$  in 1894.

One dimensional perfect arrays are often called de Bruijn [4] or Good [7] sequences. Two dimensional perfect arrays are called also perfect maps [16] or de Bruijn tori [8–10].

De Bruijn sequences of even length—introduced in [11]—are useful in construction of perfect arrays when the size of the alphabet is an even number and the window size is  $2 \times 2$ . Their definition is as follows.

**Definition 7.** If  $n$  is an even integer then an  $(n, 1, 2, n^2)$ -perfect sequence  $M = (m_1, m_2, \dots, m_{n^2})$  is called *even*, if  $m_i = x$ ,  $m_{i+1} = y$ ,  $x \neq y$ ,  $m_j = y$  and  $m_{j+1} = x$  imply  $j - i$  is even.

Iványi and Tóth [11] and later Hurlbert and Isaak [9] provided a constructive proof of the existence of even sequences.

**Definition 8.** Lexicographic indexing of an array  $M = [m_{j_1 j_2 \dots j_d}] = [m_{\mathbf{j}}] (1 \leq j_i \leq b_i)$  for  $i \in [1..d]$  means that the index  $I(m_{\mathbf{j}})$  is defined as

$$I(m_{\mathbf{j}}) = j_1 - 1 + \sum_{i=2}^d \left( (j_i - 1) \prod_{m=1}^{i-1} b_m \right).$$

The concept of perfectness can be extended to infinite arrays in various ways. In *growing arrays* [9] the window size is fixed, the alphabet size is increasing and the prefixes grow in all  $d$  directions.

**Definition 9.** Let  $a$  and  $d$  be positive integers with  $a \geq 2$  and  $\mathbf{n} = \langle n_1, n_2, \dots \rangle$  be a strictly increasing sequence of positive integers. An array  $M = [m_{i_1 i_2 \dots i_d}]$  is called  $(\mathbf{n}, d, a)$ -growing, if the following conditions hold:

- (1)  $M = [m_{i_1 i_2 \dots i_d}]$  ( $1 \leq i_j < \infty$ ) for  $j \in [1..d]$ ;

- (2)  $m_{i_1 i_2 \dots i_d} \in [0..n-1]$ ;  
 (3) the prefix  $M_k = [m_{i_1 i_2 \dots i_d}]$  ( $1 \leq i_j \leq n_k^{a^d/d}$  for  $j \in [1..d]$ ) of  $M$  is  $(n_k, d, a, n_k^{a^d/d})$ -perfect array for  $k \in [0..\infty]$ .

For the growing arrays we use the terms growing sequence, growing square and growing cube.

**Definition 10.** For  $a, n \in [2..\infty]$  the new alphabet size  $N(n, a)$  is

$$N(n, a) = \begin{cases} n & \text{if any prime divisor of } a \text{ divides } n, \\ nq & \text{otherwise,} \end{cases} \quad (1)$$

where  $q$  is the product of the prime divisors of  $a$  not dividing  $n$ .

Note, that alphabet size  $n$  and new alphabet size  $N$  have the property that  $n \mid N$ , furthermore,  $n = N$  holds in the most interesting case  $d = 3$  and  $n = a_1 = a_2 = a_3 = 2$ .

The aim of this paper is to prove the existence of a double cube. As a side-effect we show that there exist  $(n, d, a)$ -growing matrices for any  $n, d$  and  $a$ .

## 2. Necessary condition and earlier results

Since in the period  $M$  of a perfect array  $A$  each element is the head of a pattern, the volume of  $M$  equals the number of the possible patterns. Since each pattern—among others the pattern containing only zeros—can appear only once, any size of  $M$  is greater than the corresponding size of the window. So we have the following necessary condition [2,9]: If  $M$  is an  $(n, d, \mathbf{a}, \mathbf{b})$ -perfect array, then

$$|\mathbf{b}| = n^{|\mathbf{a}|} \quad (2)$$

and

$$b_i > a_i \quad \text{for } i \in [1..d]. \quad (3)$$

Different construction algorithms and other results concerning one and two dimensional perfect arrays can be found in the fourth volume of *The Art of Computer Programming* written by Knuth [12]. For example, a  $(2, 1, 5, 32)$ -perfect array [12, p. 22], a 36-length even sequence whose 4-length and 16-length prefixes are also even sequences [12, p. 62], a  $(2, 2, 2, 4)$ -perfect array [12, p. 38] and a  $(4, 2, 2, 16)$ -perfect array [12, p. 63].

It is known [4,12] that in the one-dimensional case the necessary condition (2) is sufficient too. There are many construction algorithms, like the ones of Cock [2], Fan et al. [5], Martin [14] or any algorithm for constructing of directed Euler cycles [13].

Chung et al. [1] posed the problem to give a necessary and sufficient condition of the existence of  $(n, 2, \mathbf{a}, \mathbf{b})$ -perfect arrays.

The conditions (2) and (3) are sufficient for the existence of  $(2, 2, \mathbf{a}, \mathbf{b})$ -perfect arrays [5] and  $(n, 2, a, b)$ -perfect arrays [15]. Paterson in [16] supplied further sufficient conditions.

Hurlbert and Isaak [9] gave a construction for one and two dimensional growing arrays.

## 3. Algorithms for constructing growing de Bruijn arrays

In the construction of perfect de Bruijn arrays we use the following algorithms.

Algorithm MARTIN [14] generates de Bruijn sequences. Its inputs are the alphabet size  $n$  and the window size  $a$ . Its output is an  $n$ -ary perfect sequence of length  $n^a$ . The output begins with  $a$  zeros and always continues with the maximal permitted element of the alphabet.

Algorithm EVEN [9] produces even de Bruijn sequences.

Algorithm MESH [9,11] produces doubly symmetric cellular perfect arrays when  $n$  is even,  $d = 2$ ,  $a_1 = 2$  and  $a_2 = 2$ . The input of algorithm MESH is an even alphabet size  $n$  and an even de Bruijn sequence  $e_1, e_2, \dots, e_{n^2}$ , the output is

an  $(n, 2, n^2, n^2)$ -perfect array  $P$ , whose elements are calculated by the meshing function [11]:

$$P_{ij} = \begin{cases} e_j & \text{if } i + j \text{ is even,} \\ e_i & \text{if } i + j \text{ is odd,} \end{cases} \quad (4)$$

Algorithm SHIFT [2] is a widely usable algorithm to construct perfect arrays. We use it to transform cellular  $(N, d, a, \mathbf{b})$ -perfect arrays into  $(N, d + 1, a, \mathbf{c})$ -perfect arrays.

We introduce three new algorithms.

CELLULAR results cellular perfect arrays. Its input data are  $n$ ,  $d$  and  $\mathbf{a}$ , its output is an  $(N, d, \mathbf{a}, \mathbf{b})$ -perfect array, where  $b_1 = N^{a_1}$  and  $b_i = N^{a_1 a_2 \dots a_i - a_1 a_2 \dots a_{i-1}}$  for  $i = 2, 3, \dots, d$ . CELLULAR consists of five parts:

- (1) *Calculation* (line 1 in the pseudocode) determining the new alphabet size  $N$  using formula (1);
- (2) *Walking* (lines 2–3) if  $d = 1$ , then construction of a perfect symmetric sequence  $S_1$  using algorithm MARTIN (walking in a de Bruijn graph);
- (3) *Meshing* (lines 4–6) if  $d = 2$ ,  $N$  is even and  $a = 2$ , then first construct an  $N$ -ary even perfect sequence  $\mathbf{e} = \langle e_1, e_2, \dots, e_{N^2} \rangle$  using EVEN, then construct an  $N^2 \times N^2$  sized  $N$ -ary square  $S_1$  using meshing function (4);
- (4) *Shifting* (lines 7–12) if  $d > 1$  and ( $N$  is odd or  $a > 2$ ), then use MARTIN once, then use SHIFT  $d - 1$  times, receiving a perfect array  $P$ ;
- (5) *Combination* (lines 13–16) if  $d > 2$ ,  $N$  is even and  $a = 2$ , then construct an even sequence with EVEN, construct a perfect square by MESH and finally use of SHIFT  $d - 2$  times, results a perfect array  $P$ .

COLOUR transforms cellular perfect arrays into larger cellular perfect arrays. Its input data are

- $d \geq 1$ —the number of dimensions;
- $N \geq 2$ —the size of the alphabet;
- $\mathbf{a}$ —the window size;
- $\mathbf{b}$ —the size of the cellular perfect array  $A$ ;
- $A$ —a cellular  $(N, d, \mathbf{a}, \mathbf{b})$ -perfect array.
- $k \geq 2$ —the multiplication coefficient of the alphabet;
- $\langle k_1, k_2, \dots, k_d \rangle$ —the extension vector having the property  $k^{|\mathbf{a}|} = k_1 \times k_2 \times \dots \times k_d$ .

The output of COLOUR is

- a  $(kN)$ -ary cellular perfect array  $P$  of size  $\mathbf{b} = \langle k_1 a_1, k_2 a_2, \dots, k_d a_d \rangle$ .

COLOUR consists of three steps:

- (1) *Blocking*: (line 1) arranging  $k^{|\mathbf{a}|}$  copies (blocks) of a cellular perfect array  $A$  into a rectangular array  $R$  of size  $\mathbf{k} = k_1 \times k_2 \times \dots \times k_d$  and indexing the blocks lexicographically (by  $0, 1, \dots, k^{|\mathbf{a}|} - 1$ );
- (2) *Indexing*: (line 2) the construction of a lexicographic indexing scheme  $I$  containing the elements  $0, 1, \dots, k^{|\mathbf{a}|} - 1$  and having the same structure as the array  $R$ , then construction of a colouring matrix  $C$ , transforming the elements of  $I$  into  $k$ -ary numbers consisting of  $|\mathbf{a}|$  digits;
- (3) *Colouring*: (lines 3 & 4) colouring  $R$  into a symmetric perfect array  $P$  using the colouring array  $C$  that is adding the  $N$ -fold of the  $j$ th element of  $C$  to each cell of the  $j$ th block in  $R$  (considering the elements of the cell as lexicographically ordered digits of a number).

The output  $P$  consists of blocks, blocks consist of cells and cells consist of elements. If  $e = P[\mathbf{j}]$  is an element of  $P$ , then the lexicographic index of the block containing  $e$  is called the *blockindex* of  $e$ , the lexicographic index of the cell containing  $e$  is called the *cellindex* and the lexicographic index of  $e$  in the cell is called *elementindex*. For example, the element  $S_2[7, 6] = 2$  in Table 3 has blockindex 5, cellindex 2 and elementindex 1.

Finally, algorithm GROWING generates a prefix  $S_r$  of a growing array  $G$ . Its input data are  $r$ , the number of required doubly perfect prefixes of the growing array  $G$ , then  $n$ ,  $d$  and  $\mathbf{a}$ . It consists of the following steps:

- (1) *Initialization*: construction of a cellular perfect array  $P$  using CELLULAR;

- (2) *Resizing*: if the result of the initialization is not doubly symmetric, then construction of a symmetric perfect array  $S_1$  using COLOUR, otherwise we take  $P$  as  $S_1$ ;
- (3) *Iteration*: construction of the further  $r - 1$  prefixes of the growing array  $G$  repeatedly, using COLOUR.

4. Examples of constructing growing arrays using colouring

In this section particular constructions are presented.

4.1. Construction of growing sequences

As the first example let  $n = 2$ ,  $a = 2$  and  $r = 3$ . CELLULAR calculates  $N = 2$  and MARTIN produces the cellular (2,1,2,4)-perfect sequence  $P = 00|11$ . Since  $P$  is symmetric,  $S_1 = P$ . Now GROWING chooses multiplication coefficient  $k = n_2/n_1 = 2$ , extension vector  $\mathbf{k} = \langle 4 \rangle$  and uses COLOUR to construct a 4-ary perfect sequence. COLOUR arranges  $k_1 = 4$  copies into a four blocks sized array receiving

$$R = 00|11 \parallel 00|11 \parallel 00|11 \parallel 00|11.$$

(5)

COLOURING receives the indexing scheme  $I = 0 \ 1 \ 2 \ 3$ , and the colouring matrix  $C$  transforming the elements of  $I$  into a digit length  $k$ -ary numbers:  $C = 00 \parallel 01 \parallel 10 \parallel 11$ . Finally we colour the matrix  $R$  using  $C$ —that is multiply the elements of  $C$  by  $n_1$  and adding the  $j$ th ( $j = 0, 1, 2, 3$ ) block of  $C_1 = n_1 C$  to both cells of the  $j$ th copy in  $R$ :

$$S_2 = 00|11 \parallel 02|13 \parallel 20|31 \parallel 22|33.$$

(6)

Since  $r = 3$ , we use COLOUR again with  $k = n_3/n_2 = 2$  and get the (8,1,2,64)-perfect sequence  $S_3$  repeating  $S_2$  four times, using the same indexing array  $I$  and colouring array  $C' = 2C$ . Another example is  $a = 2$ ,  $n = 3$  and  $r = 2$ . To guarantee the cellular property now we need a new alphabet size  $N = 6$ . Martin produces a (6,1,2,36)-perfect sequence  $S_1$ , then COLOUR results a (12,1,2,144)-perfect sequence  $S_2$ .

4.2. Construction of growing squares

Let  $n = a = 2$  and  $r = 3$ . Then  $N(2, 2) = 2$ . We construct the even sequence  $W_4 = e_1e_2e_3e_4 = 0 \ 0 \ 1 \ 1$  using EVEN and the symmetric perfect array  $A$  in Table 1a using the meshing function (4). Since  $A$  is symmetric, it can be used as  $S_1$ . Now the greatest common divisor of  $a$  and  $a^d$  is 2, therefore indeed  $n_1 = N^{2/2} = 2$ . GROWING chooses  $k = n_1/N = 2$  and COLOUR returns the array  $R$  repeating the array  $A$   $k^2 \times k^2 = 4 \times 4$  times.

Table 1

Column/row	1	2	3	4
(a) $A$ (2, 2, 4, 4)-square				
1	0	0	0	1
2	0	0	1	0
3	1	0	1	1
4	0	1	1	1
(b) Indexing scheme $I$ of size $4 \times 4$				
1	0	1	2	3
2	4	5	6	7
3	8	9	10	11
4	12	13	14	15

Table 2  
Binary colouring matrix  $C$  of size  $8 \times 8$

Column/row	1	2	3	4	5	6	7	8
1	0	0	0	0	0	0	0	0
2	0	0	0	1	1	0	1	1
3	0	1	0	1	0	1	0	1
4	0	0	0	1	1	0	1	1
5	1	0	1	0	1	0	1	0
6	0	0	0	1	1	0	1	1
7	1	1	1	1	1	1	1	1
8	0	0	0	1	1	0	1	1

Table 3  
A  $(4,2,2,16)$ -square generated by colouring

Column/row	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
1	0	0	0	1	0	0	0	1	0	0	0	1	0	0	0	1
2	0	0	1	0	0	2	1	2	2	0	3	0	2	2	3	2
3	1	0	1	1	1	0	1	1	1	0	1	1	1	0	1	1
4	0	1	1	1	0	3	1	3	2	1	3	1	2	3	3	3
5	0	2	0	3	0	2	0	3	0	2	0	3	0	2	0	3
6	0	0	1	0	0	2	1	2	2	0	3	0	2	2	3	2
7	1	2	1	3	1	2	1	3	1	2	1	3	1	2	1	3
8	0	1	1	1	0	3	1	3	2	1	3	1	2	3	3	3
9	2	0	2	1	2	0	2	1	2	0	2	1	2	0	2	1
10	0	0	1	0	0	2	1	2	2	0	3	0	2	2	3	2
11	3	0	3	1	3	0	3	1	3	0	3	1	3	0	3	1
12	0	1	1	1	0	3	1	3	2	1	3	1	2	3	3	3
13	2	2	2	3	2	2	2	3	2	2	2	3	2	2	2	3
14	0	0	1	0	0	2	1	2	2	0	3	0	2	2	3	2
15	3	2	3	3	3	2	3	3	3	2	3	3	3	2	3	3
16	0	1	1	1	0	3	1	3	2	1	3	1	2	3	3	3

COLOUR uses the indexing scheme  $I$  containing  $k^4$  indices in the same  $4 \times 4$  arrangement as it was used in  $R$ . Table 1b shows  $I$ .

Transformation of the elements of  $I$  into 4-digit  $k$ -ary form results the colouring matrix  $C$  represented in Table 2.

Colouring of array  $R$  using the colouring array  $2C$  results the  $(4,2,2,16)$ -square  $S_2$  represented in Table 3.

In the next iteration COLOUR constructs an 8-ary square repeating  $S_2$   $4 \times 4$  times, using the same indexing scheme  $I$  and colouring by  $4C$ . The result is  $S_3$ , a  $(8, 2, 2, 64)$ -perfect square.

#### 4.3. Construction of growing cubes

If  $d = 3$ , then the necessary condition (2) is  $b^3 = (n)^{a^3}$  for double cubes, implying  $n$  is a cube number or  $a$  is a multiple of 3. Therefore, either  $n \geq 8$  and then  $b \geq 256$ , or  $a \geq 3$  and so  $b \geq 512$ , that is, the smallest possible perfect double cube is the  $(8, 3, 2, 256)$ -cube.

Table 4  
Eight layers of a (2,3,2,16)-perfect array

Layer 0	Layer 1	Layer 2	Layer 3	Layer 4	Layer 5	Layer 6	Layer 7
0 0 0 1	0 0 0 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1
0 0 1 0	0 0 1 0	0 0 0 1	0 0 0 1	0 1 1 1	0 1 1 1	0 1 1 1	0 1 1 1
1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1
0 1 1 1	0 1 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	1 0 1 1	0 1 1 1

As an example, let  $n = 2$ ,  $a = 2$  and  $r = 2$ . CELLULAR computes  $N = 2$ , MESH constructs the  $(2, 2, 2, 4)$ -perfect square in Table 1a, then SHIFT uses MARTIN with  $N = 16$  and  $a = 1$  to get the shift sizes for the layers of the  $(2, 3, 2, \mathbf{b})$ -perfect output  $P$  of CELLULAR, where  $\mathbf{b} = \langle 4, 4, 16 \rangle$ . SHIFT uses  $P$  as zeroth layer and the  $j$ th ( $j \in [1 : 15]$ ) layer is generated by cyclic shifting of the previous layer downwards by  $w_i$  (div 4) and right by  $w_i$  (mod 4), where  $\mathbf{w} = \langle 0 \ 15 \ 14 \ 13 \ 12 \ 11 \ 10 \ 9 \ 8 \ 7 \ 6 \ 5 \ 4 \ 3 \ 2 \ 1 \rangle$ . Eight layers of  $P$  are shown in Table 4.

Let  $A_3$  be a  $4 \times 4 \times 16$  sized perfect, rectangular matrix, whose zeroth layer is the matrix represented in Table 1, and the  $(2, 3, a, b)$ -perfect array  $P$  in Table 4, where  $a = (2, 2, 2)$  and  $b = (4, 4, 8)$ .

GROWING uses COLOUR to retrieve a doubly symmetric cube.  $n_1 = 8$ , thus  $b = 256$ ,  $k = n_1/N = 4$  and  $\mathbf{k} = \langle 256/4, 256/4, 256/64 \rangle$ , that is we construct the matrix  $R$  repeating  $P$   $64 \times 64 \times 16$  times.

$I$  has the size  $64 \times 64 \times 16$  and  $I[i_1, i_2, i_3] = 64^2(i_1 - 1) + 64(i_2 - 1) + i_3 - 1$ . COLOUR gets the colouring matrix  $C$  by transforming the elements of  $I$  into 8-digit 4-ary numbers—and arrange the elements into  $2 \times 2 \times 2$  sized cubes in lexicographic order—that is in order  $(0,0,0)$ ,  $(0,0,1)$ ,  $(0,1,0)$ ,  $(0,1,1)$ ,  $(1,0,0)$ ,  $(1,0,1)$ ,  $(1,1,0)$ ,  $(1,1,1)$ . Finally colouring results a double cube  $S_1$ .

$S_1$  contains  $2^{24}$  elements therefore it is presented only in electronic form (on the homepage of the corresponding author).

If we repeat the colouring again with  $k = 2$ , then we get a 64-ary  $65536 \times 64536 \times 64536$  sized double cube  $S_2$ .

## 5. Proof of the main result

The main result of this paper can be formulated as follows.

**Theorem 11.** *If  $n \geq 2$ ,  $d \geq 1$ ,  $a \geq 2$ ,  $n_j = N^{dj/\gcd(d,a^d)}$  with  $N = N(n, a)$  given by (1) for  $j \in [0..\infty]$ , then there exists an  $(\mathbf{n}, d, a)$ -growing array.*

The proof is based on the following lemmas.

**Lemma 12 (Cellular lemma).** *If  $n \geq 2$ ,  $d \geq 1$  and  $a \geq 2$ , then algorithm CELLULAR produces a cellular  $(N, d, a, \mathbf{b})$ -perfect array  $A$ , where  $N$  is determined by formula (1),  $b_1 = N^a$  and  $b_i = N^{a^i - a^{i-1}}$  ( $i \in [2..d]$ ).*

**Proof.** It is known that algorithms EVEN + MESH and MARTIN + SHIFT result perfect outputs.

Since MESH is used only for even alphabet size and for  $2 \times 2$  sized window, the sizes of the constructed array are even numbers and so the output array is cellular.

In the case of SHIFT we exploit that all prime divisors of  $a$  divide the new alphabet size  $N$ , and  $b_i = N^{(a-1)(a^{i-1})}$  and  $(a-1)(a^{i-1}) \geq 1$ .  $\square$

**Lemma 13 (Indexing lemma).** *If  $n \geq 2$ ,  $d \geq 2$ ,  $k \geq 2$ ,  $C$  is a  $d$  dimensional  $\mathbf{a}$ -cellular array with  $|\mathbf{b}| = k^{|\mathbf{a}|}$  cells and each cell of  $C$  contains the corresponding cellindex as an  $|\mathbf{a}|$  digit  $k$ -ary number, then any two elements of  $C$  having the same elementindex and different cellindex are heads of different patterns.*

**Proof.** Let  $P_1$  and  $P_2$  be two such patterns and let us suppose they are identical. Let the head of  $P_1$  in the cell have cellindex  $g$  and head of  $P_2$  in the cell have cellindex  $h$  (both cells are in array  $C$ ). Let  $g - h = u$ .

We show that  $u = 0 \pmod{k^{|b|}}$ . For example in Table 2 let the head of  $P_1$  be (2, 2) and the head of  $P_2$  be (2, 6). Then these heads are in cells with cellindex 0 and 2 so here  $u = 2$ .

In both cells, let us consider the position containing the values having local value 1 of some number (in our example they are the elements (3,2) and (3,6) of  $C$ .) Since these elements are identical, then  $k|u$ . Then let us consider the positions with local values  $k$  (in our example they are (3,1) and (3,5).) Since these elements are also identical so  $k^2|u$ . We continue this way up to the elements having local value  $k^{|b|}$  and get  $k^{|b|}|u$ , implying  $u = 0$ .

This contradicts to the condition that the patterns are in different cells.  $\square$

**Lemma 14 (Colouring lemma).** *If  $k \geq 2$ ,  $k_i \in [2..\infty]$  ( $i \in [1..d]$ ),  $A$  is a cellular  $(n, d, \mathbf{a}, \mathbf{b})$ -perfect array, then algorithm COLOUR( $N, d, \mathbf{a}, k, \mathbf{k}, A, S$ ) produces a cellular  $(kN, d, \mathbf{a}, \mathbf{c})$ -perfect array  $P$ , where  $\mathbf{c} = \langle k_1 a_1, k_2 a_2, \dots, k_d a_d \rangle$ .*

**Proof.** The input array  $A$  is  $N$ -ary, therefore  $R$  is also  $N$ -ary. The colouring array  $C$  contains the elements of  $[0..N(k-1)]$ , so elements of  $P$  are in  $[0..kN-1]$ .

The number of dimensions of  $S$  equals to the number of dimensions of  $P$  that is,  $d$ .

Since  $A$  is cellular and  $c_i$  is a multiple of  $b_i$  ( $i \in [1..d]$ ),  $P$  is cellular.

All that has to be shown is that the patterns in  $P$  are different.

Let us consider two elements of  $P$  as heads of two windows and their contents—patterns  $p$  and  $q$ . If these heads have different cellindex, then the considered patterns are different due to the periodicity of  $R$ . For example, in Table 3  $P[11, 9]$  has cellindex 8, the pattern headed by  $P[9, 11]$  has cellindex 2, therefore they are different (see parity of the elements).

If two heads have identical cellindex but different blockindex, then the indexing lemma can be applied.  $\square$

**Proof of the main Theorem.** Lemma 18 implies that the first call of COLOUR in line 10 of GROWING results a doubly symmetric perfect output  $S_1$ . In every iteration step (in lines 14–16 of GROWING) the zeroth block of  $S_i$  is the same as  $S_{i-1}$ , since the zeroth cell of the colouring array is filled up with zeros.

Thus  $S_1$  is transformed into a doubly symmetric perfect output  $S_r$  having the required prefixes  $S_1, S_2, \dots, S_{r-1}$ .  $\square$

## 6. Final remarks

The proposed definitions and algorithms can be extended for arbitrary  $\mathbf{a}$ .

Among others, the following problems are open: existence of  $(6, 2, 5, \mathbf{b})$ -perfect array with  $\mathbf{b} = \langle 2 \times 3^8, 2^8 \times 3 \rangle$  or with  $\mathbf{b} = \langle 2 \times 3^{24}, 2^{24} \times 3 \rangle$  and the existence of a  $(2, 3, 3, 512)$ -perfect array (it would be the second smallest double cube).

## 7. Pseudocodes of the algorithms used

The algorithms are written using the pseudocode of [3]. The running time of these algorithms is determined by the number of the elements of the generated perfect array—e.g. GROWING needs  $\Theta((n_r)^{a^3})$  time.

Since we deal only with the construction of symmetric perfect arrays, the window is always symmetric.

### 7.1. Pseudocode of the algorithm GROWING

Input parameters of GROWING are  $n$ ,  $d$ ,  $a$  and  $r$ , the output is a doubly symmetric perfect array  $S_r$ , which is the  $r$ th prefix of an  $(\mathbf{n}, d, a)$ -growing array.



```

GROWING( $n, d, a, r, S_r$ )
1  CELLULAR( $n, d, a, N, P$ )
2  calculation of  $N$  using formula (1)
3  if  $P$  is symmetric then  $S_1 \leftarrow P$ 
4  if  $P$  is not symmetric then
5       $n_1 \leftarrow N^{d/\gcd(d, a^d)}$ 
6       $k \leftarrow n_1/N$ 
7       $k_1 \leftarrow (n_1)^{a^d/3}/N^a$ 
8      for  $i \leftarrow 2$  to  $d$ 
9           $k_i \leftarrow (n_1)^{a^d/d}/N^{a^i - a^{i-1}}$ 
10         COLOUR( $n_1, d, a, k, \mathbf{k}, P, S_1$ )
11  $k \leftarrow N^{d/\gcd(d, a^d)}$ 
12 for  $i \leftarrow 1$  to  $d$ 
13      $k_i \leftarrow (n_2)^{a^d/d}/N^{a^i - a^{i-1}}$ 
14 for  $i \leftarrow 2$  to  $r$ 
15  $n_i \leftarrow N^{di/\gcd(d, a^d)}$ 
16     COLOUR( $n_i, d, a, k, \mathbf{k}, S_{i-1}, S_i$ )
17 return  $S_r$ 

```

### 7.2. Pseudocode of the algorithm CELLULAR

This is an extension and combination of the known algorithms SHIFT, MARTIN, EVEN and MESH. CELLULAR( $n, d, a, N, A$ )

```

1   $N \leftarrow N(n, a)$ 
2  if  $d = 1$  then MARTIN( $N, d, a, A$ )
3  return  $A$ 
4  if  $d = 2$  and  $a = 2$  and  $N$  is even then
5      MESH( $N, a, A$ )
6      return  $A$ 
7  if  $N$  is odd or  $a \neq 2$  then
8      MARTIN( $N, a, P_1$ )
9      for  $i \leftarrow 1$  to  $d - 1$ 
10         SHIFT( $N, i, P_i, P_{i+1}$ )
11          $A \leftarrow P_1$ 
12     return  $A$ 
13 MESH( $N, a, P_1$ )
14 for  $i \leftarrow 2$  to  $d - 1$ 
15     SHIFT( $N, i, P_i, P_{i+1}$ )
16  $A \leftarrow P_d$ 
17 return  $P_d$ 

```

### 7.3. Pseudocode of the algorithm MARTIN

The following effective implementation of MARTIN is taken from [11].

MARTIN( $n, a, \mathbf{w}$ )

```

1  for  $i \leftarrow 0$  to  $n^{a-1} - 1$ 
2     $C[i] \leftarrow n - 1$ 
3  for  $i \leftarrow 1$  to  $a$ 
4     $w[i] \leftarrow 0$ 
5  for  $i \leftarrow a + 1$  to  $n^a$ 
6     $k \leftarrow w[i - a + 1]$ 
7    for  $j \leftarrow 1$  to  $a - 1$ 
8       $k \leftarrow kn + w[i - a + j]$ 
9       $w[i] \leftarrow C[k]$ 
10      $C[k] \leftarrow C[k] - 1$ 
11 return  $P$ 

```

#### 7.4. Pseudocode of the algorithm SHIFT

SHIFT( $N, d, a, P_d, P_{d+1}$ )

```

1  MARTIN( $N^{a^d}, a - 1, \mathbf{w}$ )
2  for  $j \leftarrow 0$  to  $N^{a^d - a^{d-1}} - 1$ 
3    transform  $w_i$  to an  $a^d$  digit  $N$ -ary number
4    produce the  $(j + 1)$ -st layer of the output  $P_{d+1}$  by multiple shifting the  $j$ th layer of  $P_d$  by the transformed
    number (the first  $a$  digits give the shift size for the first direction, then the next  $a^2 - a$  digits in the second
    direction etc.)
5  return  $P_{d+1}$ 

```

#### 7.5. Pseudocode of the algorithm EVEN

If  $N$  is even, then this algorithm generates the  $N^2$ -length prefix of an even growing sequence [9].

EVEN( $N, \mathbf{w}$ )

```

1  if  $N = 2$  then
2     $w[1] \leftarrow 0$ 
3     $w[2] \leftarrow 0$ 
4     $w[3] \leftarrow 1$ 
5     $w[4] \leftarrow 1$ 
6    return  $\mathbf{w}$ 
7  for  $i = 1$  to  $N/2 - 1$ 
8    for  $j = 0$  to  $2i - 1$ 
9       $w[4i^2 + 2j + 1] \leftarrow j$ 
10     for  $j = 0$  to  $i - 1$ 
11        $w[4i^2 + 2 + 4j] \leftarrow 2i$ 
12     for  $j = 0$  to  $i - 1$ 
13        $w[4i^2 + 4 + 4j] \leftarrow 2i + 1$ 
14     for  $j = 0$  to  $4i - 1$ 
15        $w[4i^2 + 4i + 1 + j] \leftarrow w[4i^2 + 4i - j]$ 
16      $w[4i^2 + 8i + 1] \leftarrow 2i + 1$ 
17      $w[4i^2 + 8i + 2] \leftarrow 2i$ 
18      $w[4i^2 + 8i + 3] \leftarrow 2i$ 
19      $w[4i^2 + 8i + 4] \leftarrow 2i + 1$ 
20 return  $\mathbf{w}$ 

```

### 7.6. Pseudocode of the algorithm MESH

The following implementation of MESH is taken from [11].

MESH( $N, \mathbf{w}, S$ )

```

1  for  $i \leftarrow 1$  to  $N^2$ 
2      for  $j \leftarrow 1$  to  $N^2$ 
3          if  $i + j$  is even then  $S[i, j] \leftarrow w[i]$ 
4          else  $S[i, j] \leftarrow w[j]$ 
5  return  $S$ 
```

### 7.7. Pseudocode of the algorithm COLOUR

Input parameters are  $N, d, a, k, \mathbf{k}$ , a cellular  $(N, d, a, \mathbf{b})$ -perfect array  $A$ , the output is a  $(kN, d, \mathbf{a}, \mathbf{c})$ -perfect array  $P$ , where  $\mathbf{c} = \langle a_1 k_1, a_2 k_2, \dots, a_d k_d \rangle$ .

COLOUR( $N, d, \mathbf{a}, k, \mathbf{k}, A, P$ )

```

1  arrange the copies of  $P$  into an array  $R$  of size  $k_1 \times k_2 \times \dots \times k_d$  blocks
2  construct a lexicographic indexing scheme  $I$  containing the elements of  $[0..k^{a^d} - 1]$  and having the same structure as  $R$ 
3  construct an array  $C$  transforming the elements of  $I$  into  $k$ -ary numbers of  $v$  digits and multiplying them by  $N$ 
4  produce the output  $S$  adding the  $j$ th ( $j \in [0..k^{a^d} - 1]$ ) element of  $C$  to each cell of the  $j$ th block in  $R$  for each block of  $R$ 
5  return  $S$ 
```

## Acknowledgement

The authors thank Gábor Balázs and Balázs Gerőfi, students of Faculty of Informatics of Eötvös Loránd University, for constructing and validating of a  $256 \times 256 \times 256$  sized perfect double cube.

## References

- [1] F.R.K. Chung, P. Diaconis, R.L. Graham, Universal cycles for combinatorial structures, *Discrete Math.* 110 (1992) 43–59.
- [2] J.C. Cock, Toroidal tilings from de Bruijn cyclic sequences, *Discrete Math.* 70 (1988) 209–210.
- [3] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, *Introduction to Algorithms*, second ed., The MIT Press/McGraw Hill, 2003 (Fourth printing).
- [4] N.G. de Bruijn, A combinatorial problem, *Kon. Nederl. Akad. Wetensch.* 49 (1946) 758–764.
- [5] C.T. Fan, S.M. Fan, S.L. Ma, M.K. Siu, On de Bruijn arrays, *Ars Comb.* 19A (1985) 205–213.
- [6] T.M. Flye-Sainte, Solution of problem 58, *Intermédiaire des Mathématiciens* 1 (1894) 107–110.
- [7] I.J. Good, Normally recurring decimals, *J. London Math. Soc.* 21 (1946) 167–169.
- [8] G. Hurlbert, G. Isaak, On the de Bruijn torus problem, *J. Combin. Theory Ser. A* 164 (1993) 50–62.
- [9] G. Hurlbert, G. Isaak, A meshing technique for de Bruijn tori, *Contemp. Math.* 178 (1994) 153–160.
- [10] G. Hurlbert, C. Mitchell, K.G. Paterson, On the existence of the Bruijn tori with two by two window property, *J. Combin. Theory Ser. A* 76 (1996) 213–230.
- [11] A. Iványi, Z. Tóth, Existence of de Bruijn words, in: I. Peák (Ed.), *Automat. Languages Programming Syst.* 88-4 (1988) 165–172.
- [12] D. E. Knuth, *The Art of Computer Programming*, vol. 4., Fascicle 2. *Generating All Tuples and Permutations*, Addison-Wesley, Upper Saddle River, 2005.
- [13] L. Lovász, *Combinatorial Problems and Exercises*, Academic Press, Budapest, 1979.
- [14] M.H. Martin, A problem in arrangements, *Bull. American Math. Soc.* 40 (1934) 859–864.
- [15] K.G. Paterson, Perfect maps, *IEEE Trans. Inf. Theory* 40 (3) (1994) 743–753.
- [16] K.G. Paterson, New classes of perfect maps. II., *J. Combin. Theory Ser. A* 73 (1996) 335–345.